STIX-Shifter

Release 6.2.0

Md Azam, Danny Elliott

Jan 25, 2024

CONTENTS

STIX-shifter is an open source python library allowing software to connect to products that house data repositories by using STIX Patterning, and return results as STIX Observations.

STIX-Shifter github repo is the official portal of everything STIX-Shifter beyond this documentation: source, connectors, tutorial, community entrances, and more.

Overview For general information about STIX, this project, and the command line utilities, see the *STIX-shifter Overview*.

Available Connectors There are more than 30 connectors. For the list of connectors, see the *Available Connectors*.

Developer Guide Follow the developer guide to learn about developing a new STIX-Shifter connector, see *Connector Developer Guide*.

CHAPTER

ONE

INTRODUCTION

STIX-shifter is an open source python library allowing software to connect to products that house data repositories by using STIX Patterning, and return results as STIX Observations.

This library takes in STIX 2 Patterns as input, and "finds" data that matches the patterns inside various products that house repositories of cybersecurity data. Examples of such products include SIEM systems, endpoint management systems, threat intelligence platforms, orchestration platforms, network control points, data lakes, and more.

In addition to "finding" the data by using these patterns, STIX-Shifter also *transforms the output* into STIX 2 Observations. Why would we do that you ask? To put it simply - so that all of the security data, regardless of the source, mostly looks and behaves the same.

Project Documenation

For general information about STIX, this project, and the command line utilities, see the STIX-shifter Documenation

1.1 Installation

The recommended method for installing stix-shifter is via pip. Two prerequisite packages needs to be installed inlcuding the package of stix-shifter connector module to complete a stix-shifter connector installation. Run the below commands to install all the packages:

1. Main stix-shifter package: `pip install stix-shifter`
2. Stix-shifter Utility package: `pip install stix-shifter-utils`
3. Desired stix-shifter connector module package: `pip install stix-shifter-modules-<module name>` Example: `pip install stix-shifter-modules-qradar`

1.1.1 Dependencies

STIX-shifter requries Python 3.8 or greater. See the requirements file for library dependencies.

1.2 Usage

STIX-Shifter can use used the following ways:

1.2.1 As a command line utility

The STIX-Shifter comes with a bundled script which you can use to translate STIX Pattern to a native datasource query. It can also be used to translate a JSON data source query result to a STIX bundle of observable objects. You can also send query to a datasource by using a transmission option.

More details of the command line option can be found *here*

```
$ stix-shifter translate <MODULE NAME> query "<STIX IDENTITY OBJECT>" "<STIX PATTERN>" "
↪<OPTIONS>"
```

Example:

```
$ stix-shifter translate qradar query {} "[ipv4-addr:value = '127.0.0.1']" {}
```

In order to build `stix-shifter` packages from source follow the below prerequisite steps:

1. Go to the stix-shifter parent directory
2. Optionally, you can create a Python 3 virtual environemnt: `virtualenv -p python3 virtualenv && source virtualenv/bin/activate`
3. Run setup: `python3 setup.py install`

1.2.2 Running from the source

You may also use the `python3 main.py` script. All the options are the same as the command line utility described above.

Example:

```
python3 main.py translate qradar query {} "[ipv4-addr:value = '127.0.0.1']" {}
```

In order to run `python3 main.py` from the source follow the below prerequisite steps:

1. Go to the stix-shifter parent directory
2. Optionally, you can create a Python 3 virtual environemnt: `virtualenv -p python3 virtualenv && source virtualenv/bin/activate`
3. Run setup to install dependancies: `INSTALL_REQUIREMENTS_ONLY=1 python3 setup.py install`.

Note: setup.py only installs dependencies when INSTALL_REQUIREMENTS_ONLY=1 directive is used. This option is similar to `python3 generate_requirements.py && pip install -r requirements.txt`

1.2.3 As a library

You can also use this library to integrate STIX Shifter into your own tools. You can translate a STIX Pattern:

```
from stix_shifter.stix_translation import stix_translation

translation = stix_translation.StixTranslation()
response = translation.translate('<MODULE NAME>', 'query', '{}', '<STIX PATTERN>', '
↪<OPTIONS>')

print(response)
```

1.2.4 Use of custom mappings

If a connector has been installed using pip, the process for editing the STIX mappings is different than if you have pulled-down the project. When working locally, you can edit the mapping files directly. See the mapping files for the MySQL connector as an example. Editing the mapping files won't work if the connector has been installed with pip; the setup script of the stix-shifter package includes the mapppings inside `config.json`. This allows stix-shifter to injest custom mappings as part of the connector's configuration.

Refer to *Use of custom mappings* for more details on how to edit the mappings in the configuration.

CHAPTER

TWO

CONTRIBUTING

We are thrilled you are considering contributing! We welcome all contributors. Please read our guidelines for contributing.

2.1 Connector Developer Guide

2.2 CLI tools and Connector Development Labs

CHAPTER

THREE

LICENSING

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

CHAPTER

FOUR

MORE RESOURCES

4.1 Join us on Slack!

Click here and fill out the form to receive an invite to the Open Cybersecurity Alliance slack instance, then join the #stix-shifter channel, to meet and discuss usage with the team.

4.2 Introduction Webinar!

Click here to view an introduction webinar on STIX Shifter and the use cases it solves for.

4.3 Changelog

- Changelog

Table of Contents

CHAPTER

FIVE

INTRODUCTION

5.1 What is STIX?

Structured Threat Information eXpression (STIX™) is a language and serialization format that organizations can use to exchange cyber threat intelligence (CTI). CTI is represented with objects and descriptive relationships and stored as JSON for machine readability.

STIX delivers a consistent and machine-readable way to enable collaborative threat analysis, automated threat exchange, automated detection and response, and more.

To learn more about STIX, see the following references:

- Introduction to STIX
- STIX and TAXII

5.2 What is STIX-SHIFTER?

STIX-shifter is an open source python library allowing software to connect to products that house data repositories by using `STIX Patterning`, and return results as `STIX Observations`.

5.3 What is STIX Patterning? What are STIX Observations?

STIX 2 Patterning is a part of STIX that deals with the "matching things" part of STIX, which is an integral component of STIX Indicators.

5.3.1 An example of a STIX pattern:

```
[url:value = 'http://www.testaddress.com'] OR [ipv4-addr:value = '192.168.122.84']
```

A STIX Observation is the `observed-data` STIX Domain Object (SDO). You can think of this as a row of data that is returned from a search triggered by the STIX pattern, and can represent an indicator of compromise. Each observation contains one or more STIX Cyber observable Objects (SCO) which in turn has one or more properties associated to the data returned from the search.

5.3.2 An example of a STIX Observation:

```
{
  "id": "observed-data--cf2c58dc-200e-49e0-b6f7-e1997cccf707",
  "type": "observed-data",
  "created_by_ref": "identity--3532c56d-ea72-48be-a2ad-1a53f4c9c6d8",
  "objects": {
    "0": {
      "type": "network-traffic",
      "src_port": 567,
      "dst_port": 102,
      "src_ref": "1",
      "dst_ref": "2"
    },
    "1": {
      "type": "ipv4-addr",
      "value": "192.168.122.84"
    },
    "2": {
      "type": "ipv4-addr",
      "value": "127.0.0.1"
    },
    "3": {
      "type": "url",
      "value": "www.testaddress.com"
    }
  }
}
```

As anyone with experience in data science will tell you, the cleansing and normalizing of the data across domains, is one of the largest hurdles to overcome with attempting to build cross-platform security analytics. This is one of the barriers we are attempting to break down with STIX Shifter.

5.4 This sounds like Sigma, I already have that

Sigma and STIX Patterning have goals that are related, but at the end of the day has slightly different scopes. While Sigma seeks to be "for log files what Snort is for network traffic and YARA is for files", STIX Patterning's goal is to encompass *all three* fundamental security data source types - network, file, and log - and do so simultaneously, allowing you to create complex queries and analytics that span domains. As such, so does STIX Shifter. It is critical to be able to create search patterns that span SIEM, Endpoint, Network, and File levels, in order to detect the complex patterns used in modern campaigns.

5.5 What is a STIX-SHIFTER connector?

A STIX-shifter connector is a module inside Stix-Shifter library that implements an interface for:

- data source query and result set translation
- data source communication

Developing a new connector expands on the data sources that STIX-shifter can support.

The combination of translation and transmission functions allows for a single STIX pattern to generate a native query for each supported data source. Each query is run, and the results are translated back into STIX objects; allowing for a uniform presentation of data.

The objective is to have all the security data, regardless of the data source to look and behave the same.

5.6 Why would I want to use this?

You might want to use this library and contribute to development, if any of the following are true:

- You are a vendor or project owner who wants to add some form of query or enrichment functions to your product capabilities
- You are an end user and want to have a way to script searches and/or queries as part of your orchestration flow
- You are a vendor or project owner who has data that can be made available, and you want to contribute a connector
- You just want to help make the world a safer place!

Take a look at the ***currently available connectors***.

CHAPTER

SIX

HOW TO USE

6.1 Prerequisites

Python 3.8 or greater is required to use stix-shifter.

Stix-shifter provides several functions: `translate` and `transmit` are the primary functions, `execute` offers a way to test the complete stix-shifter flow.

1. ***Translate***

   The `translate` command converts STIX patterns into data source queries (in whatever query language the data source might use) and translates data source results (in JSON format) into bundled STIX observation objects.

2. ***Transmit***

   The `transmit` command allows stix-shifter to connect with products that house repositories of cybersecurity data. Connection and authentication credentials are passed to the data source APIs where stix-shifter can make calls to ping the data source, make queries, delete queries, check query status, and fetch query results.

3. ***Execute***

   The translation and transmission functions can work in sequence by using the `execute` command from the CLI.

CHAPTER

SEVEN

TRANSLATE

7.1 CLI Arguments

Argument	Description	Accepted Input
TRANSLATION KEYWORD	The keyword specifiying a function will be used to translate queries and results.	`translate`
MODULE NAME	The name of the connector being used. This is the module directory name as it appears in *stix_shifter_modules*. If the connector supports multiple dialects, then by default a query will be generated for each one. You may specifiy a specific dialect by adding `:<DIALECT>` directly after the module name	The connector module name with an optional `:<DIALECT>`
TRANSLATION DATA TYPE	The type of data you wish to translate. This will be `query` for translating STIX patterns to native queries and `results` for translating data source results to STIX.	`query` or `results`
STIX IDENTITY OBJECT	This is an object that represents the data source being queried and is inserted into the results bundle of STIX objects. An empty object `"{}"` may be used for the query translation. This must be wrapped in quotes to use with the CLI.	A stringified STIX identity object
TRANSLATION DATA	This is the STIX pattern for query translation and a list of JSON results for results translation. This must be wrapped in quote to use with the CLI.	A stringified STIX pattern or list of JSON data
OPTIONS	An object of optional parameters. This must be wrapped in quotes to use with the CLI.	A stringified object of options

7.1.1 CLI Options

These are general translation options defined in `config.json` that can apply to all connectors but may be overwritten by individual modules.

Option	Translation Data Type	Description	Accepted Values
result_lim	query	The max number of results that can be returned from a query. This value is generally included in translated queries before getting sent to the data source's API query call. The default is `10000`	A number between 1 and `500000`
time_ra	query	A default time range, in minutes, applied to the translated query when no `START STOP` qualifier is present in the STIX pattern. As an example, this would be the `last x minutes` in a SQL query. The default is `5`	A number between 1 and `10000`
dialects	query	Dialects to be used for pattern translation. This will determine what `from_stix_map.json` files will be used.	A list of one or more dialect strings supported by the connector
validate_pa	query	Specifies if pattern validation is run during the query translation call. This can catch errors in the submitted STIX pattern that would otherwise raise exceptions during translation.	`true` or `false`
unmapped	results	If set to `true`, any results data returned, that is not specifired in the to-STIX mapping, will be included in the results in the following STIX object:property format `x-<MODULE NAME>:<NATIVE DATA FIELD>`. The default is `false`	`true` or `false`
stix_2.1	results	Results are returned as STIX 2.0 objects by default. Setting this option will return results in STIX 2.1 format. The default is `false`	`true` or `false`

7.2 1. Translate a STIX pattern to a native data source query

7.2.1 INPUT: STIX 2 pattern

```
[url:value = 'http://www.testaddress.com'] OR [ipv4-addr:value = '192.168.122.84']
```

Output: Native data source query

Translated Query (using SQL as an example):

```
"SELECT * FROM tableName WHERE (Url = 'http://www.testaddress.com')
OR
((SourceIpV4 = '192.168.122.84' OR DestinationIpV4 = '192.168.122.84'))"
```

7.3 CLI Command

Open a terminal and navigate to your python 3 environment. Translation of a **query** is called in the format of:

```
stix-shifter translate <MODULE NAME> query "<STIX IDENTITY OBJECT>" "<STIX PATTERN>"
"<OPTIONS>"
```

Alternatively, you can run the CLI commands from the source. Open a terminal and navigate to the stix-shifter root directory. Translation of a **query** is called in the format of:

```
python main.py translate <MODULE NAME> query "<STIX IDENTITY OBJECT>" "<STIX PATTERN>"
"<OPTIONS>"
```

The module name refers to the name of the folder in stix-shifter that contains the connector code. The current module names can be found in the *Available Connectors* table. The STIX identity object is only used when translating data source results into STIX, so it can be passed in as an empty object for query translation calls.

Using the Qradar connector as an example:

```
python main.py translate qradar query "{}" "[url:value = 'http://www.testaddress.com'] OR
[ipv4-addr:value = '192.168.122.84']"
```

7.4 Pattern translation using an input file

Create a text file with the pattern you wish to translate. The file can be used in the query translation call using standard input.

pattern.txt

```
[network-traffic:src_ref.value = '127.0.0.1'] OR [ipv4-addr:value = '0.0.0.0']
```

```
python main.py translate qradar query '{}' '' < /path/to/file/pattern.txt
```

7.5 2. Translate a JSON data source query result to a STIX 2.0 bundle of observable objects

7.5.1 INPUT: JSON data source query result

```
# Datasource results (in JSON format):
[
    {
        "SourcePort": 567,
        "DestinationPort": 102,
        "SourceIpV4": "192.168.122.84",
        "DestinationIpV4": "127.0.0.1",
        "Url": "www.testaddress.com"
    }
]
```

7.5.2 OUTPUT: STIX 2.0 bundle of observable objects

```
# STIX Observables
{
    "type": "bundle",
    "id": "bundle--2042a6e9-7f34-4a03-a745-502e358594c3",
    "spec_version": "2.0",
    "objects": [
        {
            "type": "identity",
            "id": "identity--3532c56d-ea72-48be-a2ad-1a53f4c9c6d8",
            "name": "YourDataSource",
            "identity_class": "events"
        },
        {
            "id": "observed-data--cf2c58dc-200e-49e0-b6f7-e1997cccf707",
            "type": "observed-data",
            "created_by_ref": "identity--3532c56d-ea72-48be-a2ad-1a53f4c9c6d8",
            "objects": {
                "0": {
                    "type": "network-traffic",
                    "src_port": 567,
                    "dst_port": 102,
                    "src_ref": "1",
                    "dst_ref": "2"
                },
                "1": {
                    "type": "ipv4-addr",
                    "value": "192.168.122.84"
                },
                "2": {
                    "type": "ipv4-addr",
                    "value": "127.0.0.1"
                },
                "3": {
                    "type": "url",
                    "value": "www.testaddress.com"
                }
            }
        }
    ]
}
```

7.6 CLI Command

Open a terminal and navigate to your python 3 environment. Translation of a **results** is called in the format of:

`stix-shifter translate <MODULE NAME> result '<STIX IDENTITY OBJECT>' '<LIST OF JSON RESULTS>'`

Alternatively, you can run the CLI commands from the source. Open a terminal and navigate to the stix-shifter root directory. Translation of **results** is called in the format of:

`python main.py translate <MODULE NAME> result '<STIX IDENTITY OBJECT>' '<LIST OF JSON RESULTS>'`

The module name refers to the name of the folder in stix-shifter that contains the connector code. The current module names can be found in the *Available Connectors* table. The STIX Identity object represents the data source and is passed in to allow stix-shifter to create a reference between the data source and the generated STIX observed objects.

Using the QRadar connector as an example:

```
python main.py translate qradar results \
'{"type": "identity", "id": "identity--3532c56d-ea72-48be-a2ad-1a53f4c9c6d3", "name":
↪"QRadar", "identity_class": "events"}' \
'[{"sourceip": "192.0.2.0", "filename": "someFile.exe", "sourceport": "0123", "username
↪": "root"}]'
```

7.7 Translating results into STIX 2.1

By default, JSON results are translated into STIX 2.0. To return STIX 2.1 results include `'{"stix_2.1": true}'` in the CLI command

```
python main.py translate qradar results \
'{"type": "identity", "id": "identity--3532c56d-ea72-48be-a2ad-1a53f4c9c6d3", "name":
↪"QRadar", "identity_class": "events"}' \
'[{"sourceip": "192.0.2.0", "filename": "someFile.exe", "sourceport": "0123", "username
↪": "root"}]' '{"stix_2.1": true}'
```

7.8 Validating STIX 2.0 and 2.1 bundles with the validator script

Refer to the *STIX validator*

7.9 Results translation using an input file

Create a JSON file with the results you wish to translate. The file can be used in the results translation call using standard input.

results.json

```
[
    {
        "starttime": "1563892019916",
```

(continues on next page)

(continued from previous page)

```
        "endtime": "1563892019916",
        "sourceip": "9.21.122.127",
        "sourceport": "100",
        "identityip": "0.0.0.0",
        "destinationip": "127.0.0.1",
        "destinationport": "800",
        "username": "admin",
        "protocol": "tcp"
    }
]
```

```
python main.py translate qradar results '{"type":  "identity","id":
"identity--f431f809-377b-45e0-aa1c-6a4751cae5ff","name":  "QRadar","identity_class":
"system"}' '' < /path/to/file/results.json
```

CHAPTER

EIGHT

TRANSMIT

8.1 Connection and Configuration objects

STIX-shifter expects connection and configuration objects to be passed in during transmission calls. The connection object contains the host address and port of the data source being connected to, as well as an optional self signed certificate.

8.2 Connection

This object contains information needed to connect to a specific data source. The `host` and `port` keys are required.

```
{
    "host": <Host URL or IP address>,
    "port": <Port>,
    "selfSignedCert": <false or Certificate>,
    "cert": <Certificate (if required)>,
    "resultSizeLimit": <Results limit to come back from the data source query>,
    "timeRange": <Default query time range in minutes>,
    "options": {<Any required options specific to the particular data source>}
}
```

8.3 Connection Options

These are general options defined in `config.json` that can apply to all connectors but may be overwritten by individual modules.

Option	Description	Accepted Values
timeout	The max amount of time in seconds before the query times out. The default is `30`.	A number between `1` and `60`

8.4 Configuration

This object contains an `auth` key who's value stores authentication information for the data source. What keys and values get stored in the auth will depend on the authentication requirements of the data source (username, password, auth token, etc).

```
{
    "auth": {
        "username": <Username>,
        "password": <Password>
    }
}
```

```
{
    "auth": {
        "tenant": <Tenant>,
        "clientId": <Client ID>,
        "clientSecret": <Client Secret>
    }
}
```

```
{
    "auth": {
        "SEC": <SEC Token>
    }
}
```

```
{
    "auth": {
        "token": <Security Token>
    }
}
```

```
{
    "auth": {
        "accountId": <Account ID>,
        "apiKey": <API Key>
    }
}
```

8.5 Transmit functions

Transmit offers several functions: `ping`, `query`, `status` (for asynchronous data sources), `results`, `delete` (if supported by the data source), and `is_async`.

Each of the transmit functions takes in common arguments: the module name, the connection object, and the configuration object. The module name refers to the name of the folder in stix-shifter that contains the connector code. The current module names can be found in the *Available Connectors* table. Information on the *connection and configuration objects* can also be found above. Each of the CLI commands can be run from a terminal in the stix-shifter root director.

Any failed transmission function call will return an error in the format of:

```
{'success': False, 'error': <Error message reported by API>, 'code': <Error code>}
```

8.6 CLI Arguments

Argument	Description	Accepted Input
TRANSMISSION KEYWORD	The keyword specifiying a function will be used to transmit API calls to the target data source.	`transmit`
MODULE NAME	The name of the connector being used. This is the module directory name as it appears in *stix_shifter_modules*.	The connector module name
CONNECTION OBJECT	This contains the information needed to connect to the target data source, such as host and port. This must be wrapped in quotes to use with the CLI.	A stringified connection object
CONFIGURATION OBJECT	This contains the information needed to authenticate with the target data source, such as username and password. This must be wrapped in quotes to use with the CLI.	A stringified configuration object
TRANSMISSION FUNCTION	The transmission function used to communicate with the target data source.	`is_async`, `ping`, `query`, `status`, `results`, `delete`

8.7 Transmission Functions and Arguments

Function	Description	Function Argument	Function Returns
is_as	Checks if the connector is asynchronous.	NA	`true` or `false`
ping	Calls the data source ping API endpont (or equivalent) to see if a connection can be made.	NA	Object containing `success` of `true` or `false`
query	Sends a native query string, as translated from the STIX pattern, to target data source API.	Tranlated query string	Query string
status	Checks the status of a query. Only used by asynchronous connectors.	The `search_id` returned from the query call	Object continaing `success` of `true` or `false`, `status` of `RUNNING`, `COMPLETED`, `CANCELED`, or `ERROR`, and `progress` with a number indicating the percentage complete.
results	Fetches the native results of a completed query.	The `search_id` returned from the query call followed by OFFSET and LENGTH as numbers	A list of JSON results
delete	Deletes a query from the target data source	The `search_id` returned from the query call	Object continaing `success` of `true` or `false`
results_	Fetches the results of a completed query and runs results-to-stix translation. This essentially combines the `results` transmission function with the results translation	The `search_id` returned from the query call, followed by OFFSET and LENGTH as numbers, followed by the stringified STIX identity object.	A bundle of STIX objects.

8.8 Ping

Uses the data source API to ping the connection.

8.8.1 CLI Command

```
stix-shifter transmit <MODULE NAME> '<CONNECTION OBJECT>' '<CONFIGURATION OBJECT>' ping
```

Output

```
{'success': True}
```

8.9 Query

Uses the data source API to submit a query to the connection.

8.9.1 CLI Command

```
stix-shifter transmit <MODULE NAME> '<CONNECTION OBJECT>' '<CONFIGURATION OBJECT>' query
<NATIVE DATA SOURCE QUERY>
```

Output

```
{'success': True, 'search_id': <SEARCH ID>}
```

An asynchronous data source will typically return a search ID supplied by the API response. In the event where the API doesn't return a search id, such as with a synchronous data source, the search id will be defined in the transmission module.

8.10 Status

Uses the data source API to look up the query status based on the `search_id` that is returned from the query call. This is only used for asynchronous data sources where the results are not returned right after making a query call. If the connector supports, you can specify `metadata` parameter which may contain extra information to make the status api call.

8.10.1 CLI Command

```
stix-shifter transmit <MODULE NAME> '<CONNECTION OBJECT>' '<CONFIGURATION OBJECT>' status
<SEARCH ID> <METADATA(optional)>
```

Output

```
{'success': True, 'status': <STATUS>, 'progress': <QUERY PERCENTAGE COMPLETE>}
```

The status can be one of: `COMPLETED`, `ERROR`, `CANCELLED`, `TIMEOUT`, or `RUNNING`. Depending on the data source, the progress may return with less than 100 while still showing the status as completed.

8.11 Results

Uses the data source API to fetch the query results based on the search ID, offset, and length.

If the connector supports, you can specify `metadata` parameter which may contain extra information to fetch the next batch of results from the datasource. This is a recomended parameter for the datasource that supports pagination.

8.11.1 CLI Command

```
stix-shifter transmit <MODULE NAME> '<CONNECTION OBJECT>' '<CONFIGURATION OBJECT>'
results <SEARCH ID> <OFFSET> <LENGTH> <METADATA(optional)>
```

The `OFFSET` and `LENGTH` control what pages/rows of data are returned in the query results.

Output

```
{'success': True, 'data': [<QUERY RESULTS>]}
```

Output(with metadata)

```
{'success': True, 'data': [<QUERY RESULTS>], 'metadata': <metadata values>}
```

8.11.2 Example:


```
{
    "success": true,
    "data": [
        {
            "event": {
                "securityEvent": {
                    "eventTimestamp": "2022-06-13T14:36:54.216539700Z",
                    "eventType": "FILE_CREATION",
                    "vendorName": "Microsoft",
                    "productEventType": "DeviceFileEvents",
                    "ingestedTimestamp": "2022-06-13T15:36:26.275010Z"
                },
                "securityResult": [
                    {
                        "summary": "FileCreated",
                        "category": "alert"
                    }
                ]
            }
        }
    ],
    "metadata": {
        "result_count": 2,
        "next_page_token": "CgwIlqLjoAYQ2NfggwESCwiGl52VBhC0xKB"
    }
}
```

8.12 Results as STIX

Uses the data source API to fetch the query results based on the search ID, offset, and length, and transforms into a bundle of STIX objects.

8.12.1 CLI Command

```
stix-shifter transmit <MODULE NAME> '<CONNECTION OBJECT>' '<CONFIGURATION OBJECT>'
results_stix <SEARCH ID> <OFFSET> <LENGTH> '<STIX IDENTITY OBJECT>'
```

Output

STIX bundle of objects.

The `OFFSET` and `LENGTH` control what pages/rows of data are returned in the query results.

8.13 Is Async

Checks if the data source connection is asynchronous.

8.13.1 CLI Command

```
stix-shifter transmit <MODULE NAME> '<CONNECTION OBJECT>' '<CONFIGURATION OBJECT>'
is_async
```

Output

`True` or `False`

CHAPTER

NINE

EXECUTE

The `execute` command tests all steps of the translation-transmission flow:

1. A STIX pattern is translated into a list of one or more native data source queries (using a **translate query** call).
2. Each translated query in the list is sent to the data source via a **transmit query** call.
3. If the data source is asynchronous, a **transmit status** call is made for each query. Otherwise the flow moves to the next step.
4. A **transmit results** call is made for each query (using the returned query ID in step 2). If data is returned, the resulting JSON objects get added to a list.
5. The list of JSON results get translated into a bundle of STIX objects with a **translate query** call. This bundle includes the STIX `identity` object and `observed-data` objects.

9.1 CLI Arguments

Argument	Description	Accepted Input
EXECUTE KEYWORD	The keyword specifiying that the execute function will be used.	`execute`
TRANSMISSION MODULE NAME	The name of the connector being used for transmission functions. This is the module directory name as it appears in *stix_shifter_modules*.	The connector module name
TRANSLATION MODULE NAME	The name of the connector being used for translation functions. This is the module directory name as it appears in *stix_shifter_modules*.	The connector module name
STIX IDENTITY OBJECT	This is an object that represents the data source being queried and is inserted into the results bundle of STIX objects. An empty object `"{}"` may be used for the query translation. This must be wrapped in quotes to use with the CLI.	A stringified STIX identity object
CONNECTION OBJECT	This contains the information needed to connect to the target data source, such as host and port. This must be wrapped in quotes to use with the CLI.	A stringified connection object
CONFIGURATION OBJECT	This contains the information needed to authenticate with the target data source, such as username and password. This must be wrapped in quotes to use with the CLI.	A stringified configuration object
STIX PATTERN	This is the STIX pattern to be used for the query. This must be wrapped in quote to use with the CLI.	A stringified STIX pattern

9.2 Connection Object Options

These are general options defined in `config.json` that can apply to all connectors but may be overwritten by individual modules. These should be added as an "options" objects inside the CONNECTION OBJECT.

Option	Function Type	Description	Accepted Values
result_lim	query translation	The max number of results that can be returned from a query. This value is generally included in translated queries before getting sent to the data source's API query call. The default is `10000`	A number between 1 and `500000`
time_ra	query translation	A default time range, in minutes, applied to the translated query when no `START STOP` qualifier is present in the STIX pattern. As an example, this would be the `last x minutes` in a SQL query. The default is `5`	A number between 1 and `10000`
dialects	query translation	Dialects to be used for pattern translation. This will determine what `from_stix_map.json` files will be used.	A list of one or more dialect strings supported by the connector
validate_pa	query translation	Specifies if pattern validation is run during the query translation call. This can catch errors in the submitted STIX pattern that would otherwise raise exceptions during translation.	`true` or `false`
unmapped	results translation	If set to `true`, any results data returned, that is not specifired in the to-STIX mapping, will be included in the results in the following STIX object:property format `x-<MODULE NAME>:<NATIVE DATA FIELD>`. The default is `false`	`true` or `false`
stix_2.1	results translation	Results are returned as STIX 2.0 objects by default. Setting this option will return results in STIX 2.1 format. The default is `false`	`true` or `false`
timeout	transmission	The max amount of time in seconds before the query times out. The default is `30`.	A number between 1 and `60`

9.3 CLI Command

```
stix-shifter execute <TRANSMISSION MODULE NAME> <TRANSLATION MODULE NAME> '<STIX IDENTITY
OBJECT>' '<CONNECTION OBJECT>' '<CONFIGURATION OBJECT>' '<STIX PATTERN>'
```

9.4 CLI Example

```
stix-shifter execute mysql mysql '{"type": "identity","id": "identity--f431f809-377b-45e0-aa1c-6a4751
"name": "mysql","identity_class": "system"}' '{"host": "localhost",
"database":"demo_db", "options":{"table":"demo_table", "validate_pattern": true}}'
'{"auth": {"username":"root", "password":"MyPassword"}}' "[ipv4-addr:value = '213.213.
142.5'] START t'2019-01-28T12:24:01.009Z' STOP t'2019-01-28T12:54:01.009Z'"
```

9.5 Debug

You can add the `--debug` option to your CLI command to see more logs.

```
stix-shifter --debug execute <TRANSMISSION MODULE NAME> <TRANSLATION MODULE NAME> '<STIX
IDENTITY OBJECT>' '<CONNECTION OBJECT>' '<CONFIGURATION OBJECT>' '<STIX PATTERN>'
```

9.6 Change max returned results

You can add the `--results` option with an integer value at the end of your CLI command to limit the maximum number of returned search results (default 10).

```
stix-shifter execute <TRANSMISSION MODULE NAME> <TRANSLATION MODULE NAME> '<STIX IDENTITY
OBJECT>' '<CONNECTION OBJECT>' '<CONFIGURATION OBJECT>' '<STIX PATTERN>' --results 50
```

9.7 Save the STIX results to a file

You can redirect the output of your CLI command to a file to save the STIX results.

```
stix-shifter execute <TRANSMISSION MODULE NAME> <TRANSLATION MODULE NAME> '<STIX IDENTITY
OBJECT>' '<CONNECTION OBJECT>' '<CONFIGURATION OBJECT>' '<STIX PATTERN>' > results.json
```

Output

A bundle of STIX objects

CHAPTER

TEN

MODULES

The `modules` command will return a JSON of the existing connectors along with their dialects and supported languages that are used in query translation.

10.1 CLI Command

`python main.py modules`

Output

```
{
    "qradar": {
        "flows": {
            "language": "stix",
            "default": true
        },
        "events": {
            "language": "stix",
            "default": true
        },
        "aql": {
            "language": "aql",
            "default": false
        }
    },
    "security_advisor": {
        "default": {
            "language": "stix",
            "default": true
        }
    },
    ...
}
```

This command can also be used to get the dialects of a specific connector.

`python main.py modules <module name>`

`python main.py modules qradar`

Output

```
{
    "qradar": {
        "flows": {
            "language": "stix",
            "default": true
        },
        "events": {
            "language": "stix",
            "default": true
        },
        "aql": {
            "language": "aql",
            "default": true
        }
    }
}
```

In the above example, the QRadar connector can use three dialects: `flows`, `events`, and `aql`. When a connector only has a `default` dialect, such as with Security Advisor, only one dialect is used by the connector. Most dialects will use the `stix` language since they translate STIX patterns into native queries. QRadar's `aql` dialect uses the `aql` language since it is meant to accept an AQL query rather than a STIX pattern. See the QRadar connector README for more information on AQL passthrough.

CHAPTER

ELEVEN

CONFIGS

The `configs` command returns the configuration pararmetes of the existing connectors. It basically returns a JSON of the existing connectors along with their connections and configuation objects that are specified in config.json.

11.1 CLI Command

`python main.py configs` ***Output***

```
{
    "alertflex": {
        "connection": {
            "type": {
                "type": "connectorType",
                "displayName": "Alertflex"
            },
            "options": {
                "type": "fields",
                "async_call": {
                    "type": "text",
                    "hidden": true,
                    "optional": true
                },
                "result_limit": {
                    "default": 10000,
                    "min": 1,
                    "max": 500000,
                    "type": "number",
                    "previous": "connection.resultSizeLimit"
                },
                "time_range": {
                    "default": 5,
                    "min": 1,
                    "max": 10000,
                    "type": "number",
                    "previous": "connection.timerange",
                    "nullable": true
                },
                .....
            }
        }
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Specifying the connector module name will return the configuration parameters of a specific connector.

`python main.py configs qradar`

Output


```
{
    "qradar": {
        "connection": {
            "type": {
                "type": "connectorType",
                "displayName": "IBM\u00ae QRadar and QRadar On Cloud",
                "group": "qradar"
            },
            "options": {
                "type": "fields",
                "async_call": {
                    "type": "text",
                    "hidden": true,
                    "optional": true
                },
                "result_limit": {
                    "default": 10000,
                    "min": 1,
                    "max": 500000,
                    "type": "number",
                    "previous": "connection.resultSizeLimit"
                },
                "time_range": {
                    "default": 5,
                    "min": 1,
                    "max": 10000,
                    "type": "number",
                    "previous": "connection.timerange",
                    "nullable": true
                },
                "timeout": {
                    "default": 30,
                    "min": 1,
                    "max": 60,
                    "hidden": true,
                    "type": "number",
                    "previous": "connection.timeoutLimit"
                },
                "dialects": {
                    "type": "array",
                    "hidden": true,
                    "optional": true
                },
                "language": {
```


(continues on next page)

(continued from previous page)

```
                    "type": "string",
                    "default": "stix",
                    "optional": true,
                    "hidden": true
                },
                "validate_pattern": {
                    "type": "boolean",
                    "optional": true,
                    "hidden": true,
                    "previous": "connection.validate_pattern",
                    "default": false
                },
                "stix_validator": {
                    "type": "boolean",
                    "default": false,
                    "optional": true,
                    "hidden": true,
                    "previous": "connection.stix_validator"
                },
                "mapping": {
                    "type": "json",
                    "optional": true,
                    "previous": "connection.mapping"
                },
                "unmapped_fallback": {
                    "type": "boolean",
                    "default": true,
                    "optional": true,
                    "hidden": true
                },
                "stix_2.1": {
                    "type": "boolean",
                    "default": false,
                    "optional": true,
                    "hidden": true
                }
            },
            "host": {
                "type": "text",
                "regex": "^(([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9_:/\\-]*[a-zA-Z0-9])\\.
↪)*([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9_:/\\-]*[A-Za-z0-9])$"
            },
            "port": {
                "type": "number",
                "default": 443,
                "min": 1,
                "max": 65535
            },
            "help": {
                "default": "data-sources-qradar.html",
                "type": "link"
            },
```

(continues on next page)

(continued from previous page)

```
            "selfSignedCert": {
                "type": "password",
                "optional": true
            }
        },
        "configuration": {
            "auth": {
                "type": "fields",
                "sec": {
                    "type": "password",
                    "previous": "configuration.auth.SEC"
                }
            }
        }
    }
}
```

CHAPTER

TWELVE

LIMITATIONS

STIX-Shifter has limitations on the length of a pattern that can be translated into a native query. As the pattern length increases, the translation time increases exponentially due to how ANTLR 4 parses the pattern. See STIX-Shifter Limitations for more details.

CHAPTER

THIRTEEN

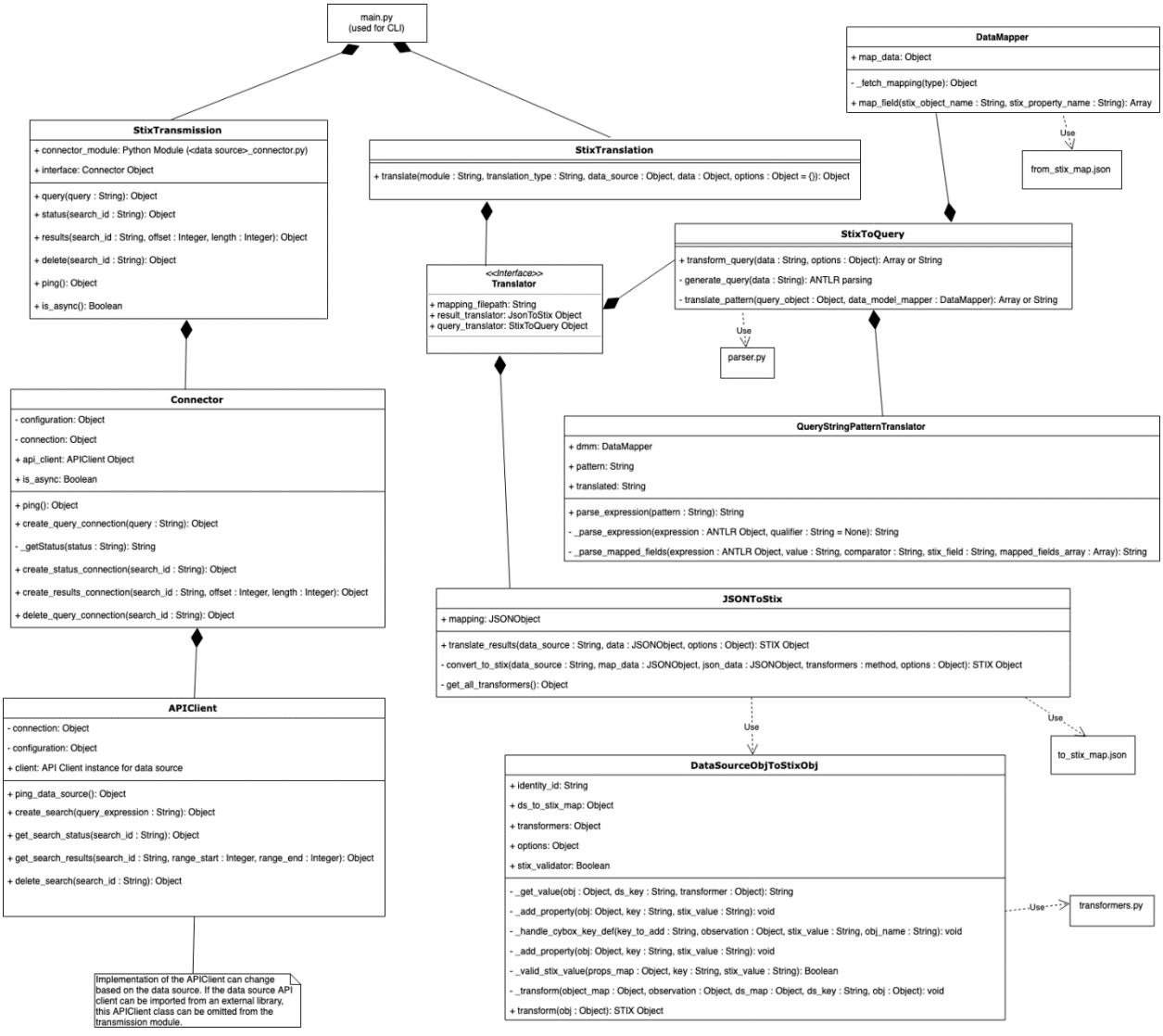
GLOSSARY

Terms	Definition
Modules	Folders in the stix-shifter project that contains code that is specific to a data source.
STIX 2 patterns	STIX patterns are expressions that represent Cyber Observable objects within a STIX Indicator STIX Domain Objects (SDOs). They are helpful for modeling intelligence that indicates cyber activity.
STIX 2 objects	JSON objects that contain CTI data. In STIX, these objects are referred to as Cyber Observable Objects.
Data sources	Security products that house data repositories.
Data source queries	Queries written in the data source's native query language.
Data source query results	Data returned from a data source query.

CHAPTER

FOURTEEN

ARCHITECTURE CONTEXT



CHAPTER

FIFTEEN

CONTRIBUTING

We are thrilled you are considering contributing! We welcome all contributors.

Please read our guidelines for contributing.

CHAPTER

SIXTEEN

GUIDE FOR CREATING NEW CONNECTORS

If you want to create a new connector for STIX-shifter, see the *developer guide*

CHAPTER

SEVENTEEN

LICENSING

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

CHAPTER

EIGHTEEN

AVAILABLE CONNECTORS

STIX-shifter currently offers connector support for the following cybersecurity products.

List updated: April 18, 2023

	Connector	Module Name	Data Model	Developer	Tra
01	IBM QRadar	qradar	QRadar AQL	IBM Security	Yes
02	IBM QRadar on Cloud	qradar	QRadar AQL	IBM Security	Yes
03	HCL BigFix	bigfix	Default	IBM Security	Yes
04	Carbon Black CB Response	carbonblack	Default	IBM Security	Yes
05	Carbon Black Cloud	cbcloud	Default	IBM Security	Yes
06	Elasticsearch	elastic	MITRE CAR	MITRE	Yes
07	Elasticsearch (ECS)	elastic_ecs	ECS	IBM Security	Yes
08	IBM Cloud Security Advisor	security_advisor	Default	IBM Cloud	Yes
09	Splunk Enterprise Security	splunk	Splunk CIM	IBM Security	Yes
10	Microsoft Defender for Endpoint	msatp	Default	IBM Security	Yes
11	Microsoft Graph Security	azure_sentinel	Default	IBM Security	Yes
12	IBM Guardium Data Protection	guardium	Default	IBM Security	Yes
13	AWS CloudWatch Logs	aws_cloud_watch_logs	Default	IBM Security	Yes
14	Amazon Athena	aws_athena	SQL	IBM Security	Yes
15	Alertflex	alertflex	Default	Alertflex	Yes
16	Micro Focus ArcSight	arcsight	Default	IBM Security	Yes
17	CrowdStrike Falcon	crowdstrike	Default	IBM Security	Yes
18	Trend Micro Vision One	trendmicro_vision_one	Default	Trend Micro	Yes
19	IBM Security Verify Privilege Vault	secretserver	Default	IBM	Yes
20	One Login	onelogin	Default	GS Lab	Yes
21	MySQL	mysql	Default	IBM	Yes
22	Sumo Logic	sumologic	Default	GS Lab	Yes
23	Datadog	datadog	Default	GS Lab	Yes
24	Infoblox BloxOne Threat Defense	infoblox	Default	Infoblox	Yes
25	Proofpoint (SIEM API)	proofpoint	Default	IBM Security	Yes
26	Cybereason	cybereason	Default	IBM Security	Yes
27	Palo Alto Cortex XDR	paloalto	Default	IBM Security	Yes
28	SentinelOne	sentinelone	Default	IBM Security	Yes
29	Darktrace	darktrace	Default	IBM Security	Yes
30	IBM Security QRadar EDR	reaqta	Default	IBM Security	Yes
31	IBM Security Verify	ibm_security_verify	Default	IBM Security	Yes
32	Red Hat Advanced Cluster Security for Kubernetes (StackRox)	rhacs	Default	IBM Security	Yes
33	GCP Chronicle	gcp_chronicle	Default	IBM Security	Yes
34	Azure Log Analytics	azure_log_analytics	Default	IBM Security	Yes

Table 1 – continued from previous page

	Connector	Module Name	Data Model	Developer	Tra
35	Okta	okta	Default	IBM Security	Yes

CHAPTER

NINETEEN

DEVELOPING A NEW CONNECTOR

19.1 Scenario

Participants

This scenario involves a software developer (*Developer A*) and an end user (*User A*). *Developer A* wants to implement a new connector for the STIX-shifter project that can support a particular security product (*Product A*). *User A* is another developer that uses the STIX-shifter library.

Problem to solve

User A performs security monitoring with *Product A* and several other security products. The other products already have existing STIX-shifter connectors.

User A would like to:

1. Submit one STIX pattern to query all the user's security products at once. The use of a STIX pattern simplifies the search process because *User A* does not need to know the query language or API calls for each security product.
2. See the query results from all the security products in one unified format (STIX bundle). With the assumption that the submitted pattern represents a potential security incident, the STIX bundle presents the query results in the context of the security event.

By implementing a new connector, *Developer A* allows *Product A* to fit into the workflow.

19.2 Prerequisites

- Your development environment must use Python 3.8 or greater.
- You must have access to the target data source. In the sample scenario, you must have access to Product A data source.
- You must be familiar with Product A's query language and APIs.
- You must be familiar or understand the following concepts:
  - Observable objects. See STIX™ Version 2.0. Part 4: Cyber Observable Objects
  - Stix patterning. See STIX™ Version 2.0. Part 5: STIX Patterning

19.3 Best practices

Familiarize yourself with some *best practices* before beginning a new connector.

19.4 Steps

To develop a STIX-shifter connector for a data source:

1. Fork the `opencybersecurityalliance/stix-shifter` repository from https://github.com/opencybersecurityalliance/stix-shifter to work on your own copy of the library.
2. *Create a module folder*.
3. *Create a Translation module*.
4. *Create a Transmission module*.
5. *Create Configuration JSONs*.
6. *Create the module entry points*.
7. Create a pull request to merge your changes in the `opencybersecurityalliance/stix-shifter` repository.

19.4.1 Create a module folder

Connector modules are stored under the `stix_shifter_modules` directory. To help you get started with creating a new connector, two module templates are available. If your data source executes queries synchronously (there is no API call to check the status of the query), make a copy of the `synchronous_template` folder in the `stix_shifter_modules` directory. If your data source executes queries asynchronously, make a copy of the `async_template` folder. The instructions that follow use the async template as an example.

Rename the copied folder to match the data source your new connector is being developed for. For example, `abc_security_monitor`.

The module name is used as an argument when either translation or transmission is called. This argument is used throughout the project so that STIX-shifter knows which modules to use.

Each module contains the following directories and files:

stix_translation: Directory containing files needed for STIX translation.

stix_transmission: Directory containing files for executing API calls to run data source queries.

configuration: Directory containing configuration files.

entry_point.py: Initializes classes and paths used by the connector.

19.4.2 Create the module entry points

The `EntryPoint` class acts as a gateway to the various methods used by the translation and transmission classes. In most instances, it's fine to use the `setup_transmission_simple` and `setup_translation_simple(dialect_default='default')` methods. In cases where multiple dialects are used by the connector, the `dialect_default` argument is the dialect you wish to use as the default when the entire collection isn't passed in. See *Create a Translation module* to learn about dialects.

19.4.3 Entry points for synchronous connections

If the data source is synchronous, you must include `set_async(False)` in the connector's entry point initialization, otherwise the data source will be treated as asynchronous by default. Even though a synchronous connector omits the `query_connector.py` and `status_connector.py` files from its transmission directory, those connectors are still called in every module's entry point. For synchronous data sources, those connectors call the `BaseSyncConnector()` methods which return `{"success": True, "status": "COMPLETED", "progress": 100}`.

```
class EntryPoint(EntryPointBase):
  def __init__(self, connection={}, configuration={}, options={}):
    super().__init__(connection, configuration, options)
    self.set_async(False)
    if connection:
      api_client = APIClient(connection, configuration)
      base_sync_connector = BaseSyncConnector()
      ping_connector = PingConnector(api_client)
      query_connector = base_sync_connector
      status_connector = base_sync_connector
      ...
      self.set_query_connector(query_connector)
      self.set_ping_connector(ping_connector)
      self.set_status_connector(status_connector)
      self.set_delete_connector(delete_connector)
      self.set_results_connector(results_connector)
    else:
      ...
```

19.4.4 Testing a new connector using the proxy host

Work on a new stix-shifter connector occurs after the project has been forked and cloned into a local development environment. Stix-shifter contains a **proxy** connector that facilitates a remote instance of the project calling out to a local instance. While in development, a new connector's working branch can be tested in any project using the stix-shifter library without first merging into the master branch on Github. A host is run on the local instance from the CLI. When a `proxy` data source is passed to the remote instance of stix-shifter, the real connection attributes (data source type, host, and port contained in the options) are passed onto the local instance of stix-shifter running the proxy host. The host will then use the new connector and return results back to the remote stix-shifter instance.

Open a terminal and navigate to your local stix-shifter directory. Run the host with the following command:

```
python main.py host "<STIX Identity Object>" "<Host IP address>:<Host Port>"
```

As an example:

```
python main.py host '{"type": "identity","id": "identity--f431f809-377b-45e0-aa1c-
↪6a4751cae5ff","name": "Bundle","identity_class": "events"}' "192.168.122.83:5000"
```

Calling the proxy host

Each of the translate and transmit CLI commands outlined in the stix-shifter overview can be used to call the proxy host.

As an example:

```
python main.py transmit proxy '{"options": {"proxy_host": "127.0.0.1", "proxy_port":␣
↪5000, "destination": {"connection": {"options": {"result_limit": 10000, "time_range":␣
↪5, "timeout": 30}, "host": "<HOST>", "port": <PORT>, "type": "qradar"}, "configuration
↪": {"auth": { "SEC": "<SEC TOKEN>"} } } }}' '{}' ping
```

19.4.5 Packaging individual connectors

Stix-shifter can be broken into several python whl packages by using the `setup.py` script found in the root of the project. This packaging script can be called from the CLI:

```
MODE='<module name>' VERSION='<connector version>' python3 setup.py bdist_wheel
```

`MODE` is a required argument that is used to determine how the project is packaged. Mode options include:

`'1'` = Include everything in one whl package

`'3'` - 3 whl packages respectively for stix-shifter, stix-shifter-utils and stix-shifter-modules

`'N'` - `stix-shifter`, `stix-shifter-utils`, and each connector is packaged separately

`<module name>` - package only the specified connector

The `VERSION` argument is optional. If missing, version 1.0.0 is attached to the package name.

When the script is executed, a new `dist` directory is created at the root of the stix-shifter project; this contains the generated whl packages.

A packaged connector follows the naming convention of:

`stix_shifter_modules_<module name>-<version>-py2.py3-none-any.whl`

The contents of the package has the same directory structure as the module in the project:

```
stix_shifter_modules =>
    <module name> =>
        configuration
        stix_translation
        stix_transmission
        entry_point
```

19.4.6 Building images of the connectors

You can build the docker image your developed connector locally and publish it to your desired repository. In order to do that, follow the below steps-

1. Make sure you have built the wheel distribution of the connector module by following the steps in *Packaging individual connectors section*.
2. `image_builder` directory in your stix-shifter project contains the required scripts that will automatically build the connector image. You can copy the directory in a separate location or keep it inside stix-shifter project.

3. Create a folder named `bundle` inside `image_builder/` directory.
4. Move your desired connector wheel file (`stix_shifter_modules_<module name>-<version>-py2.py3-none-any.whl`) to the bundle folder created in step 3.
5. Run `build_local.sh` script.
6. Image should be automatically built in your running docker client.

CHAPTER

TWENTY

STIX TRANSLATION

The steps below assume you have renamed the `async_template` module directory to our example connector name, `abc_security_monitor`.

1. *Exploring the stix_translation directory*
2. *Edit the from_stix_map.json file*
3. *Edit the operators.json file*
4. *Edit the query_constructor.py file*
5. *Edit the to_stix_map.json file*

   Refer to the *keywords document* when creating the to-STIX mappings.
6. *Add custom data transformers (optional)*
7. *Verify that the translation module was created successfully*

20.1 Step 1. Exploring the stix_translation directory

Verify that your `stix_translation` directory contains the following folders and files.

Folder/file	Why is it important? Where is it used?
json/<DIALE(	These mapping files are used to translate a STIX pattern to a data source query result.
json/to_stix_m	This mapping file is used to translate a data source query result into STIX objects.
operators.json	This file maps the STIX comparison and observation operators with the associated data source operators.
__init__.py	This file is required by Python to properly handle library directories.
query_constru(	This file contains the QueryStringPatternTranslator class, which translates the ANTLR parsing of the STIX pattern to the native data source query.
query_translat(	This file contains the QueryTranslator class, which inherits the BaseQueryTranslator class. QueryTranslator calls out to the ANTLR parser, which returns a parsing of the STIX pattern. The parsing is then passed onto the query_constructor.py where it is translated into the native data source query.
results_translato	This file contains the ResultsTranslator class, which inherits the JSON ToStix class.
transformers.py (optional)	This optional file will contain any connector-specific data transformers that are not included in the shared `stix_shifter_utils/stix_translation/src/utils/transformers.py` file.

Back to top

20.2 Step 2. Edit the from_stix_map JSON files

The <DIALECT>_from_stix_map.json files is where you define HOW to translate a STIX pattern to a native data source query. STIX patterns are expressions that represent Cyber Observable objects. The mapping of STIX objects and properties to data source fields determine how a STIX pattern is translated to a data source query.

1. Identify your data source fields.
2. Refer to the following documentation STIX™ Version 2.0. Part 4: Cyber Observable Objects for the list of STIX objects that you can map your data source fields.
3. Edit the `from_stix_map.json` file found in the `json` directory. The `dialect1_from_stix_map.json` file contains a sample mapping of STIX objects and properties to data source fields in the following format:

```
{
    "stix-object": {
        "fields": {
            "stix_object_property": ["DataSourceField", "DataSourceField"],
            "stix_object_property": ["DataSourceField"]
        }
    }
}
```

4. Map your data source fields to a STIX object and property. Define the mapping based on the specified format. You can map multiple data source fields to the same STIX object property.
   - "stix-object" refer to a STIX cyber observable object type name
   - "stix_object_property" refers to a STIX cyber observable object property name

Example mapping

The following example illustrates the mapping of STIX objects (network-traffic, ipv4-addr, and url) to a data source with the fields – SourcePort, DestinationPort, StartTime, EndTime, NetworkProtocol, SourceIpV4, DestinationIpV4, and Url.

```
{
    "network-traffic": {
        "fields": {
          "src_port": ["SourcePort"],
          "dst_port": ["DestinationPort"],
          "start": ["StartTime"],
          "end": ["EndTime"],
          "protocols[*]": ["NetworkProtocol"]
        }
      },
      "ipv4-addr": {
        "fields": {
          "value": ["SourceIpV4", "DestinationIpV4"]
        }
      },
      "url": {
          "fields": {
              "value": ["Url"]
          }
      }
}
```

The following STIX pattern is supported in the example mapping because the STIX objects (network-traffic and ipv4-addr) and their properties are defined in the file and mapped to data source fields.

```
"[network-traffic:src_port = 12345 AND ipv4-addr:value = '00-00-5E-00-53-00']"
```

Using multiple from-STIX map files with dialects

Pattern translation can use dialects to differentiate between multiple from-STIX mapping files. Multiple from-STIX mappings may be needed in cases where one STIX pattern queries multiple data source tables that use different schemas. Any dialects are appended to the module name with the following format: `<module_name>:<dialect_1>:<dialect_2>` Using QRadar as an example, one pattern queries both event and flow tables. This requires a from-STIX mapping file for each, which results in one pattern translating into two AQL queries. QRadar's module name would be passed to the `StixTranslation.translate` method as `qradar:events:flows`. Each dialect gets extracted from the module name and is used throughout the pattern translation flow. In cases where multiple from-STIX map files are used, the naming convention is `<dialect>_from_stix_map.json`. It is important that the file names follow this structure since the dialect is used to dynamically look up the file path. So in the case of QRadar, there would be a `events_from_stix_map.json` and `flows_from_stix_map.json` file in the json folder. The dialect can also be used in the `query_constructor` (detailed below) if it's needed in the translated query string. In the case of an SQL language, this may look like `SELECT * FROM <dialect> WHERE <some condition>`

If your data source uses multiple dialects, rename the `<DIALECT>_from_stix_map.json` files to include the dialect at the beginning of the file name. Include as many mapping files as needed; one for each dialect. If your data source only uses one dialect, include only one from-STIX mapping file with the name `from_stix_map.json` in the json directory.

Back to top

Using Custom STIX Objects and Properties

As shown below in *Step 4*, custom objects and properties are supported by the STIX standard and can be used when defining mappings. However, services using stix-shifter may be unaware of any custom elements and so only rely on standard STIX objects when constructing queries. Therefore it is recommended to stick to standard objects and attributes (as outlined in the STIX documentation) when constructing the `from_stix_map`.

20.3 Step 3. Edit the operators.json file

The `operators.json` file maps the STIX pattern operators to the data source query operators. Change the comparator values to match the operators supported in your data source.

The default data source operators are ones used in an SQL query.

Example:

```
{
    "ComparisonExpressionOperators.And": "AND",
    "ComparisonExpressionOperators.Or": "OR",
    "ComparisonComparators.GreaterThan": ">",
    "ComparisonComparators.GreaterThanOrEqual": ">=",
    "ComparisonComparators.LessThan": "<",
    "ComparisonComparators.LessThanOrEqual": "<=",
    "ComparisonComparators.Equal": "=",
    "ComparisonComparators.NotEqual": "!=",
    "ComparisonComparators.Like": "LIKE",
    "ComparisonComparators.In": "IN",
    "ComparisonComparators.Matches": "LIKE",
    "ComparisonComparators.IsSubSet": "SUBSET",
```

(continues on next page)

(continued from previous page)

```
    "ComparisonComparators.IsSuperSet": "SUPERSET",
    "ObservationOperators.Or": "OR",
    "ObservationOperators.And": "OR"
}
```

STIX pattern operators	Data source query operators
ComparisonExpressionOperators.And	And
ComparisonExpressionOperators.Or	OR
ComparisonExpressionOperators.GreaterThan	>
ComparisonExpressionOperators.GreaterThanOrEqual	>=
ComparisonExpressionOperators.LessThan	<
ComparisonComparators.LessThanOrEqual	<=
ComparisonComparators.Equal	=
ComparisonComparators.NotEqual	!=
ComparisonComparators.Like	LIKE
ComparisonComparators.In	IN
ComparisonComparators.Matches	LIKE

20.4 Step 4. Edit the query constructor file

When a STIX pattern is translated by STIX-shifter, it is first parsed with ANTLR 4 into nested expression objects. The native data source query is constructed from these nested objects.

The following STIX pattern:

```
"[network-traffic:src_port = 37020 AND network-traffic:dst_port = 635] START '2016-06-
↪01T00:00:00Z' STOP '2016-06-01T01:11:11Z'"
```

Translates into the following ANTLR parsing:

```
Pattern[
    ObservationExpression(
        CombinedComparisonExpression(
            ComparisonExpression(
                network-traffic:dst_port ComparisonComparators.Equal 635
            )
            ComparisonExpressionOperators.And
            ComparisonExpression(
                network-traffic:src_port ComparisonComparators.Equal 37020
            )
        )
    )
    Qualifier(
        STARTt'2016-06-01T00:00:00Z'STOPt'2016-06-01T01:11:11Z'
    )
]
```

The parsing is recursively run through QueryStringPatternTranslator._parse_expression, which is found in `query_constructor.py`.

The `query_constructor.py` file is where the native query is built from the ANTLR parsing.

20.4.1 How STIX-shifter handles unmapped STIX properties and operators

If a STIX pattern contains an unmapped property or operator, and any joining operators logically allow for it, that portion of the parsing is removed from the ANTLR objects. The modified ANTLR parsing is then transformed into one or more native queries by the query constructor. Looking at the following examples:

`[stix_object:unmapped_property OR stix_object:mapped_property]`

The unmapped property would be removed since it is joined to a mapped property with an OR operator.

`[stix_object:unmapped_property] OR [stix_object:mapped_property]`

The entire observation object (square brackets) containing the unmapped property would be removed since the pattern contains at least one other observation with mapped properties.

If the unmapped property cannot be removed, STIX-shifter produces an error. This happens because the `QueryStringPatternTranslator` class (in `query_constructor.py`) does not know what data source field the STIX object property must be converted to. The following patterns would produce such an error:

`[stix_object:unmapped_property AND stix_object:mapped_property]`

The unmapped property cannot be removed without changing the query logic since the two properties are joined by an AND operator.

`[stix_object:unmapped_property]`

The pattern only contains one observation with one unmapped property; nothing would be left to the query after removing it.

Edit the `query_constructor.py` file in the translation directory. Update the following sections based on the requirements of your data source.

20.4.2 1. Define the _parse_expression method

The ANTLR parsing is recursively run through the _parse_expression method. The type of expression is determined on each iteration. When the expression is a ComparisonExpression, a query string is added to the final data source query.

This image illustrates where the query string is constructed for the data source query.

```
def _parse_expression(self, expression, qualifier=None) -> str:
    if isinstance(expression, ComparisonExpression):  # Base Case
        # Resolve STIX Object Path to a field in the target Data Model
        stix_object, stix_field = expression.object_path.split(':')

        … final return of query string
```

The following ComparisonExpression from an ANTLR parsing:

```
ComparisonExpression(
   network-traffic:src_port ComparisonComparators.Equal 37020
)
```

Would add the following string to the native query: `"SourcePort = 37020"`

20.4.3 2. Define the final query that gets returned in the translate_pattern method

Depending on your data source, edit this section to:

- Add a query field selector.
- Append result limits and time windows.
- Return a list of one or more queries. A list is returned because some query languages require the STIX pattern to be split into multiple query strings.

The example provided in the template connector is based on an SQL language. This should to be changed to fit with the native data source query language. Each string in the return list is a query that will be passed to the data source's API via the STIX transmission `<module>_connector.py`. If the data source does not use a query language, API end points and parameters could be defined here instead (in conjunction with `<DIALECT>_from_stix_map.json`).

Back to top

20.5 Step 5. Edit the to_stix_map JSON file

The `to_stix_map.json` file is where you define HOW to translate data source query results into a bundle of STIX objects. Query results must be in JSON format; otherwise, the data source cannot be supported. There are keywords which need to be specified in the to-stix mappings in order to perform specific operations on the datasource fields. To understand the keywords and their usage, see *To STIX mapping Keywords*

Results from unmapped data source fields are ignored during translation and are not included in the bundle.

1. Identify your data source fields.
2. Refer to the following documentation STIX™ Version 2.0. Part 4: Cyber Observable Objects for the list of STIX objects that you can map your data source fields.
3. Edit the `to_stix_map.json` file in the `translation/json` directory. The `to_stix_map.json` file contains a sample mapping of data source fields to STIX objects and properties in the following format:

```
{
    "DataSourceField": {
        "key": "stix-object.stix_object_property"
    }
}
```

4. Each JSON object in the mapping has a "key" element with a value that represents a STIX object and its property. Define the mapping based on the specified format.
    - `stix-object` refer to a STIX cyber observable object type name
    - `stix_object_property` refers to a STIX cyber observable object property name

Example mapping Using the same data source as in step 3, the following example shows a to-STIX mapping:

```
{
    "Url": {
        "key": "url.value"
    },
    "SourcePort": {
        "key": "network-traffic.src_port",
        "object": "nt",
        "transformer": "ToInteger"
    },
    "DestinationPort": {
        "key": "network-traffic.dst_port",
        "object": "nt",
        "transformer": "ToInteger"
    },
    "SourceIpV4": [
        {
          "key": "ipv4-addr.value",
          "object": "src_ip"
        },
        {
          "key": "network-traffic.src_ref",
          "object": "nt",
          "references": "src_ip"
        }
      ],
    "DestinationIpV4": [
        {
            "key": "ipv4-addr.value",
            "object": "dst_ip"
        },
        {
            "key": "network-traffic.dst_ref",
            "object": "nt",
            "references": "dst_ip"
        }
    ],
    "NetworkProtocol": {
        "key": "network-traffic.protocols",
        "object": "nt",
        "transformer": "ToLowercaseArray"
    },
    "EventAction": {
        "key": "x-oca-event.action"
    },
    "EventTime": [
        {
            "key": "created"
        },
        {
            "key": "modified"
        },
        {
            "key": "first_observed"
```

(continues on next page)

(continued from previous page)

```
        },
        {
            "key": "last_observed"
        }
    ]
}
```

About the example mapping

- Url is a simple mapping.
- SourcePort and DestinationPort
  - Have matching "object" values, which cause the src_port and dst_port to be added to the same object (in this case, network-traffic).
  - Uses the ToInteger transformer. Transformers are optional mapping attributes that apply a transformation method to the data before it is written to the STIX object. The shared transform methods are in `stix_shifter_utils/stix_translation/src/utils/transformers.py`. Any new transformer classes must be added to the module's own `stix_shifter_modules/<module>/stix_translation/transformers.py` file.
- SourceIpV4 and DestinationIpV4 contain two objects.
  - The first object creates an ipv4-addr object for each of the values. Given the field, the "object" property is set to either src_ip or dst_ip.
  - The second object in the mapping adds references in the network-traffic object to the ipv4-addr objects. Since the second part of the mappings has the object set to "nt", the references are added to the same network-traffic object that contains the source and destination ports.
- NetworkProtocol
  - Mapped similarly to the source and destination ports.
  - Note the use of the ToLowercaseArray transformer. The example data source returns a single string in the NetworkProtocol field. However, in STIX, network-traffic protocols store an array of protocols in lowercase format.
- LogSourceId
  - An example of a custom STIX property.
  - Custom properties allow for data that would not fit in any existing STIX object type to be added to the observed-data object. Custom properties must start with an x_. In this example, the data source name is used as the custom object name and log_source is the custom property.
- EventTime
  - Data source field indicating the time of the event.
  - Maps to the following STIX fields: created, modified, first_observed, last_observed

Example observed-data STIX object

Input data

Url	SourcePort	DestinationPort	SourceIpV4	DestinationIpV4	NetworkProtocol	LogSourceId	EventTime
www.example	3000	1000	192.0.2.0	198.51.100.0	TCP	678	2019-04-24T12:44:00.605Z

Output

The following illustrates an observed-data STIX object that is derived from the previous example mapping and sample input data.


```
{
      "id": "observed-data--6ecb744f-37d2-4950-a7bb-9dc821679c52",
      "type": "observed-data",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2019-04-24T12:44:00.605Z",
      "first_observed": "2019-04-24T12:44:00.605Z",
      "last_observed": "2019-04-24T12:44:00.605Z",
      "modified": "2019-04-24T12:44:00.605Z",
      "number_observed": 1,
      "objects": {
        "0": {
          "type": "ipv4-addr",
          "value": "192.0.2.0"
        },
        "1": {
          "type": "network-traffic",
          "src_ref": "0",
          "src_port": 3000,
          "dst_ref": "2",
          "dst_port": 1000
        },
        "2": {
          "type": "ipv4-addr",
          "value": "198.51.100.0"
        },
        "3": {
          "type": "url",
          "value": "www.example.com"
        }
      },
      "x_my_data_source": {
        "log_source": 678
      }
    }
```


Required fields:

Every STIX observed-data object must include the following properties:

Property	Description
id	Object ID. Created automatically during results translation.
type	Object type. This will always be `observed-data` and is created automatically during results translation.
created_by_re	References the identity object that represents the data source. Created automatically during results translation.
created	Timestamp when the observed-data object was written to STIX. Created automatically during results translation.
modified	Timestamp when the current observed-data STIX object was last modified. This will often be the same value as the created property and is created automatically during results translation.
first_observ	Timestamp when the event that created the observed-data object was first observed. Must be defined in the to-STIX mapping.
last_observ	Timestamp when the event that created the observed-data object was last observed. Must be defined in the to-STIX mapping.
number_observ	The number of the same events that get returned in the results. Can be defined in the to-STIX mapping if an event count is returned in the results, otherwise this should have a default of 1.

The code for translating data source results to STIX is found in `stix_shifter_utils/stix_translation/src/json_to_stix/json_to_stix_translator.py`. Normally, there is no need to edit this file.

20.5.1 Using multiple to-STIX map files with dialects

Query results translation can use dialects to differentiate between multiple to-STIX mapping files. Multiple to-STIX mappings may be needed in cases where datasource returns multiple tables that use different schemas. Any dialects are appended to the module name with the following format: `<module_name>:<dialect_1>:<dialect_2>` Using AWS Athena as an example, datasource can return multiple schemas such as OCSF, VPC Flow and Guardduty. This requires a to-STIX mapping file for each. When the datasource returns query results for a specific schema then the appropriate to-STIX mapping file can be used based on the dialect specified in the query. Dialects can be specified in the CLI as `aws-athena:ocsf` or in the connection object as-

```
{
  "connection": {
    options: {
      dialects: ['ocsf']
    }
  }
}
```

Each dialect gets extracted from the CLI module name or the connection object and is used throughout the pattern translation and results translation flow. In cases where multiple to-STIX map files are used, the naming convention is `<dialect>_to_stix_map.json`. It is important that the file names follow this structure since the dialect is used to dynamically look up the file path. So in the case of AWS Athena, there would be a `ocsf_to_stix_map.json`, `vpcflow_to_stix_map.json` and `guardduty_to_stix_map.json` file in the json folder.

If your data source uses multiple dialects, rename the `<DIALECT>_to_stix_map.json` files to include the dialect at the beginning of the file name. Include as many mapping files as needed; one for each dialect. If your data source only uses one dialect, include only one to-STIX mapping file with the name `to_stix_map.json` in the json directory. Alternatively, you can also create one large `to_stix_map.json` that combines all the datasource fields from different schemas instead of multiple to-STIX mapping files.

Back to top

20.6 Step 6. Add custom data transformers (optional)

Sometimes data returned in the native results needs to be reformatted before getting added to a STIX object. For example, STIX may require a different timestamp format from what is returned by the data source. Transformer classes are used in these cases. The main `transformers.py` file is located in `stix_shifter_utils/stix_translation/src/utils/transformers.py`. It contains the shared classes that transform data formats. Each class has a method that takes in data and transforms it into the preferred format.

These classes can be used in cases such as:

- When converting from STIX, the data source query language requires specific data formats.
- When converting to STIX, the STIX object requires specific data formats. In this case, the format of a value that is returned in the data source results must be transformed during translation into a bundle of STIX objects. See STIX™ Version 2.0. Part 4: Cyber Observable Objects for STIX data formats.

If a connector requires a new transformer specific to the target data source, the class must be added to the `transformers.py` file within the module's `stix_translation` folder. If a transformers.py file doesn't exist in this folder, create a new one.

Back to top

20.7 Step 7. Verify that the translation module was created successfully

You must have access to the data source either through a UI or CLI so that you can run the translated query and confirm that it works. The translation module can be tested by calling the `main.py file` from the command line and passing in the required arguments. The order of arguments is as follows:

```
python main.py translate <data source (module) name> <"query" or "result"> <STIX␣
↪identity object> <pattern or JSON results to
be translated> <options>
```

20.7.1 Test the STIX pattern to data source query translation

1. Run the translation module from the command line. For example, using `abc_security_monitor` as a data source:

```
python main.py translate abc_security_monitor query '{}' "[network-traffic:src_port =␣
↪37020 and network-traffic:dst_port = 635] OR [ipv4-addr:value = '333.333.333.0'] AND␣
↪[url:value = 'www.example.com'] START t'2019-01-28T12:24:01.009Z' STOP t'2019-01-
↪28T12:54:01.009Z'"
```

To run validation on the STIX pattern, add `'{"validate_pattern": "true"}'` as an option to the end of the CLI command:

```
python main.py translate abc_security_monitor query '{}' "[network-traffic:src_port =␣
↪37020 and network-traffic:dst_port = 635] OR [ipv4-addr:value = '333.333.333.0'] AND␣
↪[url:value = 'www.example.com'] START t'2019-01-28T12:24:01.009Z' STOP t'2019-01-
↪28T12:54:01.009Z'" '{"validate_pattern": "true"}'
```

If the translation module uses multiple from-STIX mapping files, you can append the dialects to the module name before passing it into the CLI. Given that module abc_security_monitor has two dialects, it could be passed in as:

abc_security_monitor:dialect_1:dialect_2 Only the appended dialects will be used during pattern translation. abc_security_monitor:dialect_2 would only use the mapping for the second dialect and thus only return one translated query, not two. If the module uses dialects, but only the module name is passed in, all dialects will automatically be used; so abc_security_monitor would still use both dialects.

2. Visually verify the returned query by running it against the data source.

20.7.2 Test the JSON data source results to STIX translation

1. Run the translation module from the command line. For example, using abc_security_monitor as a data source:

```
python main.py translate abc_security_monitor results '{"type": "identity","id":
↪"identity--f431f809-377b-45e0-aa1c-
6a4751cae5ff", "name": "abc_security_monitor", "identity_class": "events"}' '[{"Url":
↪"www.example.com", "SourcePort": 3000, "DestinationPort": 1000, "SourceIpV4": "192.0.2.
↪0", "DestinationIpV4": "198.51.100.0", "NetworkProtocol": "TCP"}]'
```

By default, STIX 2.0 results will be returned. Adding the {"stix_2.1": true} option to the end of the CLI command will return STIX 2.1 objects. For example:

```
python main.py translate abc_security_monitor results '{"type": "identity","id":
↪"identity--f431f809-377b-45e0-aa1c-
6a4751cae5ff", "name": "abc_security_monitor", "identity_class": "events"}' '[{"Url":
↪"www.example.com", "SourcePort": 3000, "DestinationPort": 1000, "SourceIpV4": "192.0.2.
↪0", "DestinationIpV4": "198.51.100.0", "NetworkProtocol": "TCP"}]' '{"stix_2.1": true}'
```

You may validate both STIX 2.0 and 2.1 results with the Bundle validator script.

2. Visually verify that all expected data is in the returned STIX bundle. If a data source field in your sample results is mapped in to_stix_map.json, the value must be in the STIX bundle under the mapped STIX property.

Note:

- The <STIX identity object> represents a data source and is the first observed-data object that gets added to the STIX bundle during results translation.
- Each observed-data object, which gets added to the bundle, references the <STIX identity object> to indicate which data source the result came from.
- The <STIX identity object> is only used when translating data source results to STIX. As such, an empty JSON object can be passed in when converting a STIX pattern to a data source query.
- For more information about identity objects, see the STIX 2 documentation.

Back to top

CHAPTER

TWENTYONE

USE OF CUSTOM MAPPINGS

Follow the below steps, if a user or threat hunter wants to use custom mapping after installing stix-shifter libraries:

1. Go to the standard python library installation location. The installation path usually looks like this ***lib/pythonX.Y/site-packages*** or go to https://docs.python.org/3/install/ for more details on the python library installation based on your system.
2. Go to the ***stix_shifter_modules*** folder and find the connector name that is installed.
3. Inside the connector folder, go to the ***config.json*** file found under the ***stix_shifter_modules/<CONNECTOR>/configuration/*** directory.
4. There is a `mapping` object nested inside the `options` JSON object. This includes all the mappings from the `from_stix` and `to_stix` mapping files. Here's an example of the `config.json` file:

```
{
    "connection": {
        "type": {
            "displayName": "MySQL",
            "group": "mysql",
            "type": "connectorType"
        },
        "options": {
            "mapping": {
                "type": "json",
                "optional": true,
                "previous": "connection.mapping",
                "default": {
                    "from_stix_map": {
                        "ipv4-addr": {
                            "fields": {
                                "value": [
                                    "source_ipaddr",
                                    "dest_ipaddr"
                                ]
                            }
                        },
                        "file": {
                            "fields": {
                                "name": [
                                    "filename"
                                ]
                            }
```

(continues on next page)

(continued from previous page)

```
                        }
                    },
                    "operators": {
                        "ComparisonExpressionOperators.And": "AND",
                        "ComparisonExpressionOperators.Or": "OR"
                    },
                    "to_stix_map": {
                        "source_ipaddr": [
                            {
                                "key": "ipv4-addr.value",
                                "object": "src_ip"
                            },
                            {
                                "key": "network-traffic.src_ref",
                                "object": "nt",
                                "references": "src_ip"
                            }
                        ],
                        "dest_ipaddr": [
                            {
                                "key": "ipv4-addr.value",
                                "object": "dst_ip"
                            },
                            {
                                "key": "network-traffic.dst_ref",
                                "object": "nt",
                                "references": "dst_ip"
                            }
                        ]
                    }
                }
            }
        }
    },
    "configuration": {
        "auth": {
            "type": "fields",
            "username": {
                "type": "password"
            },
            "password": {
                "type": "password"
            }
        }
    }
}
```

5. You can change, update or use the existing custom mappings fields and save the file.
6. The stix-shifter CLI commands should automatically pick up your custom mappings in the next command execution.

CHAPTER

TWENTYTWO

MAPPING KEYWORDS

There are keywords which need to be specified in the `to-stix` mappings in order to perform specific operations on the datasource fields. There are two types of keywords:

1. Required
2. Optional

The below table contains the keywords and their usages:

22.1 Required Keywords

```
{
  "sha256hash": {
    "key": "file.hashes.SHA-256",
    "object": "fl"
  }
}
```

```
{
  "sourceip": {
    "key": "ipv4-addr.value",
    "object": "src_ip"
  }
}
```

22.2 Optional Keywords

22.3 Examples of Optional keywords:

22.3.1 unwrap

Mapping:

```
  "resolved_ip": [
      {
        "key": "ipv4-addr.value",
```

(continues on next page)

(continued from previous page)

```
        "object": "resolved_ip",
        "unwrap": true
      }
    ]
}
```

Datasource Result:

```
{
  "resolved_ip": [
    "40.116.120.16", "1.2.3.4"
  ]
}
```

STIX Translation

This STIX bundle contains two ipv4-addr objects which are created based on `unwrap` keyword:

```
{
    "type": "bundle",
    "id": "bundle--f3b77b73-f21f-49b8-be6b-6034b47f5b60",
    "objects": [
        {
            "type": "identity",
            "id": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
            "name": "elastic_ecs",
            "identity_class": "events",
            "spec_version": "2.0",
            "created": "2022-03-23T14:15:56.519Z",
            "modified": "2022-03-23T14:15:56.519Z"
        },
        {
            "id": "observed-data--ad31fb85-7723-4923-bb68-fa52e101e9b9",
            "type": "observed-data",
            "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
            "created": "2023-07-20T14:36:18.711Z",
            "modified": "2023-07-20T14:36:18.711Z",
            "objects": {
                "0": {
                    "type": "ipv4-addr",
                    "value": "40.116.120.16"
                },
                "1": {
                    "type": "ipv4-addr",
                    "value": "1.2.3.4"
                }
            },
            "first_observed": "2019-04-21T11:05:07.000Z",
            "last_observed": "2019-04-21T11:05:07.000Z",
            "number_observed": 1
        }
    ],
    "spec_version": "2.0"
```

(continues on next page)

(continued from previous page)

```
}
```

22.3.2 group

Mapping:

```
{
  "sourceip": [
    {
      "key": "ipv4-addr.value",
      "object": "host_ip"
    },
    {
      "key": "x-oca-asset.ip_refs",
      "object": "host",
      "references": ["host_ip"],
      "group": true
    }
  ],
  "identityip": [
    {
      "key": "ipv4-addr.value",
      "object": "host_ip_addr_v4"
    },
    {
      "key": "x-oca-asset.ip_refs",
      "object": "host",
      "references": ["host_ip"],
      "group": true
    }
  ]
}
```

Datasource Result:

```
{
    "identityip": "127.0.0.1",
    "sourceip": "10.10.10.10",
    "identityhostname": "host.com"
}
```

STIX Translation

`ip_refs` STIX property contains two reference objects which is grouped together in a list when `group` keyword is used:

```
{
    "type": "bundle",
    "id": "bundle--8d3b18d9-cbc4-4788-83e7-dd1e6a9026c9",
    "objects": [
        {
            "type": "identity",
```

(continues on next page)

(continued from previous page)


```
            "id": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
            "name": "qradar",
            "identity_class": "events",
            "spec_version": "2.0",
            "created": "2022-03-23T14:15:56.519Z",
            "modified": "2022-03-23T14:15:56.519Z"
        },
        {
            "id": "observed-data--9b7896ba-7a1a-4417-a61b-61b15b017721",
            "type": "observed-data",
            "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
            "created": "2023-07-20T18:06:32.907Z",
            "modified": "2023-07-20T18:06:32.907Z",
            "objects": {
                "0": {
                    "type": "ipv4-addr",
                    "value": "127.0.0.1"
                },
                "1": {
                    "type": "x-oca-asset",
                    "ip_refs": [
                        "0",
                        "3"
                    ],
                    "hostname": "host.com"
                },
                "3": {
                    "type": "ipv4-addr",
                    "value": "10.10.10.10"
                }
            },
            "first_observed": "2023-07-20T18:06:32.907Z",
            "last_observed": "2023-07-20T18:06:32.907Z",
            "number_observed": 1
        }
    ],
    "spec_version": "2.0"
}
```


22.3.3 group_ref

Mapping:

A custom field needs to be created to use the `group_ref` keyword. The name of the field can be anything. Make sure the mapping is defined under same nested dictionary as datasource fields. In this example, `groupReference` is the custom field. The reference object is `target` hence `groupReference` is placed under `"target":{}`. The `x_target_refs` property will store the references of `target` objects in `x-oca-event` object. You must specify `"group_ref": true` in the mapping for `groupReference` custom field.


```
{
  "eventType": {
```


(continues on next page)

(continued from previous page)

```
    "key": "x-oca-event.action",
    "object": "event"
  },
  "target": {
    "id": {
      "key": "x-okta-target.target_id",
      "object": "target"
    },
    "type": {
      "key": "x-okta-target.target_type",
      "object": "target"
    },
    "groupReference": {
      "key": "x-oca-event.x_target_refs",
      "object": "event",
      "references": [
        "target"
      ],
      "group_ref": true
    }
  }
}
```

Datasource Result:

"target" datasrouce field contains nested dictionaries. The above mapping will create two `x-okta-target` objects and a `x-oca-event` object from the below datasource result.

```
{
    "eventType": "user.authentication.auth_via_mfa",
    "target": [
        {
            "id": "00u7rkrly9sNvp7sa5d7",
            "type": "User",
            "alternateId": "user1@login.com",
            "displayName": "user1"
        },
        {
            "id": "pfd7rkr4nqHLoMqI85d7",
            "type": "AuthenticatorEnrollment",
            "alternateId": "unknown",
            "displayName": "Okta Verify",
        }
    ]
}
```

STIX Translation

Two `x-okta-target` objects(1 and 2) are referenced in `x_target_refs` property inside `x-oca-event` object when `group_ref` keyword is used in the mapping.

```
{
    "id": "observed-data--c0b44436-3f99-4d39-ade0-509c65e990d4",
```

(continues on next page)

(continued from previous page)


```
    "type": "observed-data",
    "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
    "created": "2023-11-29T18:16:13.340Z",
    "modified": "2023-11-29T18:16:13.340Z",
    "objects": {
        "0": {
            "type": "x-oca-event",
            "action": "user.authentication.auth_via_mfa",
            "x_target_refs": [
                "1",
                "2"
            ]
        },
        "1": {
            "type": "x-okta-target",
            "target_id": "00u7rkrly9sNvp7sa5d7",
            "target_type": "User"
        },
        "2": {
            "type": "x-okta-target",
            "target_id": "pfd7rkr4nqHLoMqI85d7",
            "target_type": "AuthenticatorEnrollment"
        }
    },
    "first_observed": "2023-11-29T18:16:13.340Z",
    "last_observed": "2023-11-29T18:16:13.340Z",
    "number_observed": 1
}
```


22.3.4 value

Mapping:

```
{
  "event": {
    "original": [
      {
        "key": "artifact.payload_bin",
        "transformer": "ToBase64",
        "object": "artifact"
      },
      {
        "key": "artifact.mime_type",
        "object": "artifact",
        "value" : "text/plain"
      }
    ]
  }
}
```

Datasource Result:

```
{
          "@timestamp": "2019-04-21T11:05:07.000Z",
          "event": {
            "original": "10.42.42.42 - - [07/Dec/2018:11:05:07 +0100] \"GET /blog HTTP/1.
↪1\" 200 2571 \"-\" \"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_0) AppleWebKit/537.
↪36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36\""
          }
}
```

STIX Translation

`mime_type` value has been set from the mapping `value` keyword:

```
{
    "id": "observed-data--fb592d78-942b-4829-9a3e-aacb14f9eb27",
    "type": "observed-data",
    "created_by_ref": "identity--3532c56d-ea72-48be-a2ad-1a53f4c9c6d3",
    "created": "2023-07-20T19:08:18.458Z",
    "modified": "2023-07-20T19:08:18.458Z",
    "objects": {
        "0": {
            "type": "artifact",
            "payload_bin":
↪"MTAuNDIuNDIuNDIgLSAtIFswNy9EZWMvMjAxODoxMTowNTowNyArMDEwMF0gIkdFVCAvYmxvZyBIVFRQLzEuMSIgMjAwIDI1NzEg
↪",
            "mime_type": "text/plain"
        }
    },
    "first_observed": "2019-04-21T11:05:07.000Z",
    "last_observed": "2019-04-21T11:05:07.000Z",
    "number_observed": 1
}
```

22.3.5 references

Mapping:

```
{
  "sourceip": [
    {
      "key": "ipv4-addr.value",
      "object": "src_ip"
    },
    {
      "key": "network-traffic.src_ref",
      "object": "nt",
      "references": "src_ip"
    }
  ],
  "protocol": {
    "key": "network-traffic.protocols",
    "object": "nt"
```

(continues on next page)

(continued from previous page)

```
  }
}
```

Datasource Result:

```
    "sourceip": "10.10.10.10",
    "protocol": "TCP"
}
```

STIX Translation

Source `ipv4-addr` object number is referenced in `network-traffic` object:

```
{
    "type": "bundle",
    "id": "bundle--7c70d70e-e6a1-4e31-8f21-78efee48737a",
    "objects": [
        {
            "type": "identity",
            "id": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
            "name": "qradar",
            "identity_class": "events",
            "spec_version": "2.0",
            "created": "2022-03-23T14:15:56.519Z",
            "modified": "2022-03-23T14:15:56.519Z"
        },
        {
            "id": "observed-data--f353936e-ec99-4975-b0c3-498b22bf10fb",
            "type": "observed-data",
            "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
            "created": "2023-07-21T13:37:32.811Z",
            "modified": "2023-07-21T13:37:32.811Z",
            "objects": {
                "0": {
                    "type": "ipv4-addr",
                    "value": "10.10.10.10"
                },
                "1": {
                    "type": "network-traffic",
                    "src_ref": "0",
                    "protocols": [
                        "tcp"
                    ]
                }
            },
            "first_observed": "2023-07-21T13:37:32.811Z",
            "last_observed": "2023-07-21T13:37:32.811Z",
            "number_observed": 1
        }
    ],
    "spec_version": "2.0"
}
```

22.3.6 transformer

Mapping:

```
{
  "sourceip": [
    {
      "key": "ipv4-addr.value",
      "object": "src_ip"
    },
    {
      "key": "network-traffic.src_ref",
      "object": "nt",
      "references": "src_ip"
    }
  ],
  "protocol": {
    "key": "network-traffic.protocols",
    "object": "nt"
  },
  "sourceport": {
    "key": "network-traffic.src_port",
    "object": "nt",
    "transformer": "ToInteger"
  }
}
```

Datasource Result:

```
{
    "sourceip": "10.10.10.10",
    "protocol": "TCP",
    "sourceport": "3000"
}
```

STIX Translation

Port value is transformed from string to integer when `ToInteger` transformer is set in the mapping:

```
{
    "type": "bundle",
    "id": "bundle--0aee4703-bf5b-4830-9a4a-de29c8b526fd",
    "objects": [
        {
            "type": "identity",
            "id": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
            "name": "qradar",
            "identity_class": "events",
            "spec_version": "2.0",
            "created": "2022-03-23T14:15:56.519Z",
            "modified": "2022-03-23T14:15:56.519Z"
        },
        {
            "id": "observed-data--9d80b67b-b2df-49a7-b16a-5f197b98d437",
```

(continues on next page)

(continued from previous page)


```
            "type": "observed-data",
            "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
            "created": "2023-07-21T13:54:25.088Z",
            "modified": "2023-07-21T13:54:25.088Z",
            "objects": {
                "0": {
                    "type": "ipv4-addr",
                    "value": "10.10.10.10"
                },
                "1": {
                    "type": "network-traffic",
                    "src_ref": "0",
                    "protocols": [
                        "tcp"
                    ],
                    "src_port": 3000
                }
            },
            "first_observed": "2023-07-21T13:54:25.088Z",
            "last_observed": "2023-07-21T13:54:25.088Z",
            "number_observed": 1
        }
    ],
    "spec_version": "2.0"
}
```

CHAPTER

TWENTYTHREE

STIX TRANSMISSION

The steps below assume you have renamed the `async_template` module directory to our example connector name, `abc_security_monitor`.

{: #transmission-mod}

1. *Exploring the stix_transmission directory*
2. *Edit the apiclient.py file*
3. *Edit the template_connector.py file*
4. *Edit the template_error_mapper.py file*
5. *Verify the implementation of each transmission method*

23.1 Step 1. Exploring the stix_transmission directory

Verify that your `stix_transmission` directory contains the following folders and files.

For an asynchronous transmission module, you must have the following files:

Folder/file	Why is it important? Where is it used?
__init__.py	This file is required by Python to properly handle library directories.
api_client.py	Contains methods that make the data source API calls, used by the individual connector classes
query_connector.py	Contains class for executing a search on the data source
status_connector.py	Contains class for checking the status an active search on the data source
delete_connector.py	Contains class for deleting an active search on the data source
results_connector.py	Contains class for fetching the search results from the data source
ping_connector.py	Contains class to ping the data source
error_mapper.py	

The synchronous transmission module has no need to make status or query calls since the query is handled directly in the results API call. Therefore, the synchronous transmission module will not include `query_connector.py` or `status_connector.py`.

Back to top

23.2 Step 2. Edit the API Client

Edit the `api_client.py` APIClient class methods to make the relevant API calls to the data source.

- For a asynchronous connector, the data source API must support:
  - Pinging the data source
  - Sending a search query to the data source
  - Checking the status of a search
  - Retrieving the search results

If supported by the data source, edit the `delete_search` method, otherwise leave it as it appears in the async_template connector.

Back to top

23.3 Step 3. Edit the connector class methods

Each of the stix transmission connector classes (found in the `stix_transmission/*_connector.py` files) use `api_client.py` to make the relevant calls to the data source. Edit the class methods if required by the data source. **It is important to keep the method names and signatures as they are.** Changing them will prevent the transmission methods from working properly. You are free to add new class methods as needed.

23.3.1 Returning results in JSON format

Results from the data source need to be returned as an array of JSON objects before they can be converted into STIX. If the data source does not natively return results in this way, the `ResultsConnector.create_results_connection` method should handle any needed conversion. The results array needs to be wrapped in a string. A simple example of returned data:

```
'[{"ipaddress": "0.0.0.0", "url": "www.example.com"}, {"ipaddress": "1.1.1.1", "url":
↪"www.another.example.net"}]'
```

Note on search IDs

For asynchronous sources, the search id that gets passed into the status, delete, and results methods is the ID returned by the data source when making the query API call. This is used to keep track of the original query, allowing the status and results to be fetched. However, in the case a synchronous data source, the search id is the entire query string; this is what gets passed into the results and delete methods.

23.3.2 Returning metadata in the return object

Additional values can be returned as a metadata paremeter in the status and results return object. The data type of the `metadata` parameter can be anything based on the requirements of the connector. The recomended types are python dictionary and string. Ideal use case for this paramater is pagination query. For example, if the connector needs to store the next page token or url and the previous results count to fetch next batch of results from the datasource then the metadata can look like this:

```
{
    "result_count": 1,
    "next_page_token": "CgwImdHioAYQqKmUuQMSDAiHl52VBhD8g4"
}
```

Here's an example of the return object of the results with metadata parameter:

`{'success': True, 'data': [<QUERY RESULTS>], 'metadata': <metadata values>}`

Back to top

23.4 Step 4. Edit the error mapper file

The error mapper associates data source error codes, returned by the API, with error messages defined in the `ErrorCode` class (found in `stix_shifter_utils/utils/error_response.py`). Update the keys in the `error_mapping` dictionary to match any error codes returned by the API.

As an example, `1002: ErrorCode.TRANSMISSION_SEARCH_DOES_NOT_EXISTS` would return an error code of 'no_results' if the API returned a 1002 code. Stix-shifter returns errors in the following format:

`{'success': False, 'error': <Error message reported by API>, 'code': <Error code>}`

Back to top

23.5 Step 5. Verify each transmission method

The stix-shifter CLI can be used to test each of the transmission methods. Open a terminal on your local machine, and navigate to the root of the stix-shifter project. The format for calling a method is:

`python main.py <connector name> '<CONNECTION OBJECT>' '<CONFIGURATION OBJECT>' <METHOD NAME> '<METHOD ARGUMENTS>'`

The connection and configuration objects are explained in the transmission section of the stix-shifter *OVERVIEW.md* The objects used in the CLI commands below are just an example.

23.5.1 Test the transmission ping method.

1. Use the following CLI command:

```
python main.py transmit abc '{"host":"www.example.com", "port":1234}' '{"auth": {
↪"username": "some_user_name", "password": "some_password"}}' ping
```

2. Visually confirm that a result comes back with

```
{'success': True}
```

23.5.2 Test the transmission is_async method.

1. Use the following CLI command:

```
python main.py transmit abc '{"host":"www.example.com", "port":1234}' '{"auth": {
↪"username": "some_user_name", "password": "some_password"}}' is_async
```

2. Visually confirm that it returns true if the data source is asynchronous. Otherwise, it must return false.

23.5.3 Test the transmission query method.

1. Use the following CLI command:

```
python main.py transmit abc '{"host":"www.example.com", "port":1234}' '{"auth": {
↪"username": "some_user_name", "password": "some_password"}}' query "<Native data␣
↪source Query String>"
```

2. Visually confirm that a result comes back with

```
{'success': True, 'search_id': '<some query UUID>'}
```

3. Take note of the UUID that is returned. It is the ID to use in the rest of the tests.

23.5.4 Test the transmission status method.

1. Use the following CLI command:

```
python main.py transmit abc '{"host":"www.example.com", "port":1234}' '{"auth": {
↪"username": "some_user_name", "password": "some_password"}}' status "<Query UUID␣
↪from query test>"
```

2. Visually confirm that a result comes back with

```
{'success': True, 'status': 'COMPLETED', 'progress': 100}
```

23.5.5 Test the transmission results method.

1. Use the following CLI command:

```
python main.py transmit abc '{"host":"www.example.com", "port":1234}' '{"auth": {
↪"username": "some_user_name", "password": "some_password"}}' results "<Query UUID␣
↪from query test>" <Offset Integer> <Length Integer>
```

2. You can set the offset and length command line arguments to 1.
3. Optionally, you can set the metadata parameter if the connector supports it. Ideally used for pagination query.

```
  python main.py transmit abc '{"host":"www.example.com", "port":1234}' '{"auth": {
↪"username": "some_user_name", "password": "some_password"}}' results "<Query UUID␣
↪from query test>" <Offset Integer> <Length Integer> '<metadata>'
```

4. Visually confirm that query results are returned as JSON objects. These results can be compared to what is returned when running the query string used in test C directly on the data source API, either through a UI or the CLI.

23.5.6 Test the transmission delete method.

1. Use the following CLI command:

```
python main.py transmit abc '{"host":"www.example.com", "port":1234}' '{"auth": {
↪"username": "some_user_name", "password": "some_password"}}' delete "<Query UUID␣
↪from query test>"
```

2. Visually confirm that a result comes back with

```
{'success': True}
```

CHAPTER

TWENTYFOUR

CONFIGURATION PARAMETERS

A json file needs to be created that contains configuration parameters for each module. The configuration json file is required in order to validate the module specific parameters for a successful translation and transmission call. Please follow this naming convention when you create the file: `config.json`

A second json file is required to translate the parameters defined in `lang_en.json` for the UI. This file is necessary in order to help the UI framework show the parameters in human readable format.

24.1 File Location

Create a directory named `configuration` in your module folder. The json files mentioned above needs to be created inside `configuration`. Make sure json files saved in the following location for your new module-

```
/stix_shifter_modules/<module name>/configuration
```

24.2 JSON File Description

24.2.1 config json file

Two top level json objects needs to be preset in the file: `connection` and `configuration`. The child attributes of the `connection` object should be the parameters required for making API calls which can be used by multiple users and role levels. The `configuration` object should contain the parameters that are required for API authentication for individual users and roles.

The following example JSON contains the appropriate parameters that each module requires:

```
{
    "connection": {
        "type": {
            "default": "QRadar",
            "group": "qradar"
        },
        "host": {
            "type": "text",
            "regex": "^(([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\\-]*[a-zA-Z0-9])\\.)*([A-Za-
↪z0-9]|[A-Za-z0-9][A-Za-z0-9\\-]*[A-Za-z0-9])$"
        },
        "port": {
```

(continues on next page)

(continued from previous page)

```
            "default": 443,
            "type": "number"
        },
        "help": {
            "default": "<Help URL to configure datasource to make api call>",
            "type": "link"
        },
        "limit": {
            "default": 1000,
            "max": 10000,
            "type": "number"
        },
        "timeout": {
            "default": 1,
            "max": 60,
            "type": "number"
        },
        "cert": {
            "type": "password",
            "optional": true
        },
        "selfSignedCert": {
            "type": "password",
            "optional": true
        }
    },
    "configuration": {
        "auth": {
            "sec": {
                "type": "password"
            }
        }
    }
}
```

Each parameter in both the connection and configuration object can also have few different child attributes to define the parameter functionality. Below are the attributes that can be specified at least one or more based on the parameter function:

1. type
    - The following types can be specified for the parameters (more can be added based on data source requirements):
        - text
        - number
        - password
        - boolean
2. default
    - The default value for the parameter
3. min

- Minimum value for the parameter. If the type is text, then the value is the minimum number of characters in the value.

4. max
    - Maximum value for the parameter. If the type is text, then the value is the maximum number of characters in the value.
5. optional
    - Set this value to "true" if the parameter is optional. By default the value is "false" if not defined
6. hidden
    - Set this value to "true" if the parameter needs to be hidden by the UI. By default the value is "false" if not defined
7. regex
    - Regular expression pattern that defines what characters are permitted in the value.

Configuration object needs to have `auth` child object. `auth` object should contain the parameters that are needed for api authentication. We have put an example of qradar api authentication paramter in the above example. Here's another example of `auth` object-

```
"auth": {
    "username": {
        "type": "password"
    },
    "password": {
        "type": "password"
    }
}
```

Both connection and configuration object may contain more or different parameters than that are defined in the example above based on the individual module.

24.2.2 lang json file

The `lang_en.json` file has the similar format like `config.json`. It has different child attributes to translate the files for UI framework.

1. label
    - Label of the parameter that is visiable in the UI
2. placeholder
    - Any placeholder value that can be present in the user input field
3. description
    - Description of the parameter

Below example json is the language translation file of the above QRadar config json file:

```
{
    "connection": {
        "host": {
            "label": "Management IP address or Hostname",
            "placeholder": "192.168.1.10",
```

(continues on next page)

(continued from previous page)

```
            "description": "Specify the IP address or hostname of the data source"
        },
        "port": {
            "label": "Host Port",
            "description": "Set the port number that is associated with the host name or␣
↪IP address"
        },
        "help": {
            "label": "Need additional help?",
            "description": "More details on the datasource setting can be found in the␣
↪specified link"
        },
        "limit": {
            "label": "Result Size Limit",
            "description": "The maximum number of entries or objects that are returned␣
↪by search query.The default result size limit is 1000. The value must not be less than␣
↪1 and must not be greater than 10,000."
        },
        "timeout": {
            "label": "Query Timeout Limit",
            "description": "The time limit in minutes for how long the query is run on␣
↪the data source. The default time limit is 1. When the value is set to zero, there is␣
↪no timeout. If the query takes longer than 1 min, it is likely to indicate a problem."
        },
        "cert": {
            "label": "IBM QRadar Certificate",
            "description": "Use self-signed SSL certificate for QRadar V7.3.1 and CA␣
↪content(root and intermediate) for QRadar V7.3.2"
        }
    },
    "configuration": {
        "auth": {
            "sec": {
                "label": "Authentication Token",
                "description": "The Authentication Token is the unique identifier of the␣
↪data source that you want to establish the connection with. It is required to␣
↪authenticate the connection request."
            }
        }
    }
}
```

For easier implementation, you can copy the json files from template modules (async_template or synchronous_template) and modify according the module requirements. You can also review the configuration json files of existing modules for reference.

CHAPTER

TWENTYFIVE

BEST PRACTICES

An assessment of the data source APIs should be made before beginning implementation of a connector. The APIs should return a good coverage of cyber observable data that fits within the standard STIX `observed-data` objects. If most of the data returned is getting mapped to custom STIX objects or properties, it may be an indication that the data source is not a good fit for a connector. The APIs should also allow for robust filtering of the data, or support a query language; this is essential for executing federated searches against multiple connectors using STIX patterning.

Verify the data returned by the connector is stored in cyber observable objects as defined in the STIX 2.0 standard. Data that doesn't fit into standard cyber observable objects may be added as custom STIX objects with the following preferences:

25.1 Custom Extensions on standard STIX objects

Custom STIX may be added in user-defined custom extensions to the standard STIX objects. For example, if the data source returns fields that should be grouped together and could enrich the data presented in the file object, it could be added to the object like so:

```
"extensions": {
    "x-datasource-custom-file-extension": {
        "foo_val": "foo",
        "bar_val": "bar"
    }
}
```

See section 5.2 Custom Object Extensions in the STIX standard for more details.

25.2 Custom properties on standard STIX objects

Custom STIX can be added as a custom object property on standard STIX objects. For example if the data source returns data related to the file object but not captured in one of the file object's standard properties, it could be added like so:

```
{

    "0": {
        "type": "file",
        "name": "myfile.exe",
        "x_datasource_custom_property": "bar"
```

(continues on next page)

(continued from previous page)

```
    }
}
```

25.3 Custom STIX objects

Custom STIX can be added in a custom object with the type following the naming convention of `x-datasource-object-type`

25.4 Other considerations

Data should not be repeated in the same observed-data object. That is, if a data element is represented as a property on a standard STIX object, it should not also be included as a custom property. If a custom STIX object needs to refer to an existing property on another object, it should do so using referencing rather than repeating the data.

Ensure blank data values are not written to the STIX results. This should happen automatically via the json-to-STIX flow but this may need to be improved to prevent bad data from slipping through (ie. certain IP values and mac addresses that we wish to strip out).

When a connector is close to completion, generate a sample STIX bundle with the *execute command* and run it through the *validator script* to catch errors.